간단한 선형 회귀 수행

In [1]:
```
import numpy as np # 데이터과학을 위한 패키지
from sklearn import linear_model  # scikit-learn 모듈을 가져온다

regr = linear_model.LinearRegression()
```

In [2]:
```
X = [[163], [179], [166], [169], [171]] # 독립변수 2차원리스트
y = [54, 63, 57, 56, 58] # 종속변수 1차원 리스트
regr.fit(X, y) # 학습시ㅣㅋ기
```

Out[2]:
```
▾ LinearRegression
LinearRegression()
```

In [3]:
```
print(regr.get_params())
```

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'positive': False}
```

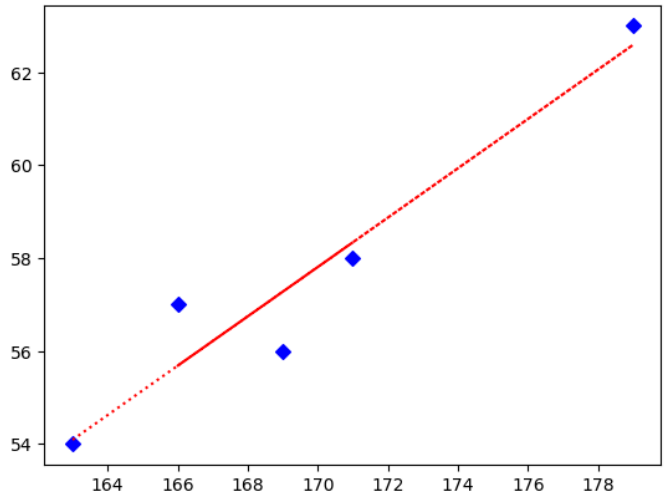
In [4]:
```
coef = regr.coef_            # 직선의 계수
intercept = regr.intercept_  # 직선의 절편
score = regr.score(X, y)     # 학습된 직선이 데이터를 얼마나 잘 따르나

print("y = {}* X + {:.2f}".format(coef.round(2), intercept))
print("데이터와 선형회귀 직선의 관계점수: {:.1%}".format(score)) #
```

```
y = [0.53]* X + -32.50
데이터와 선형회귀 직선의 관계점수: 91.9%
```

데이터 시각화 및 차원 증가

In [5]:
```
import matplotlib.pyplot as plt

# 학습 데이터와 y 값을 산포도로 그린다.
plt.scatter(X, y, color='blue', marker='D')
# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = regr.predict(X) # 예측 함수
# 계산된 기울기와 y 절편을 가지는 점선을 그려보자
plt.plot(X, y_pred, 'r:')
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x189df6bcc10>]
```



In [9]:
```
unseen = [[167]]
result = regr.predict(unseen)
print('동윤이의 키가 {}cm 이므로 몸무게는 {}kg으로 추정됨'.format(W
                    unseen, result.round(1)))
```

```
동윤이의 키가 [[167]]cm 이므로 몸무게는 [56.2]kg으로 추정됨
```

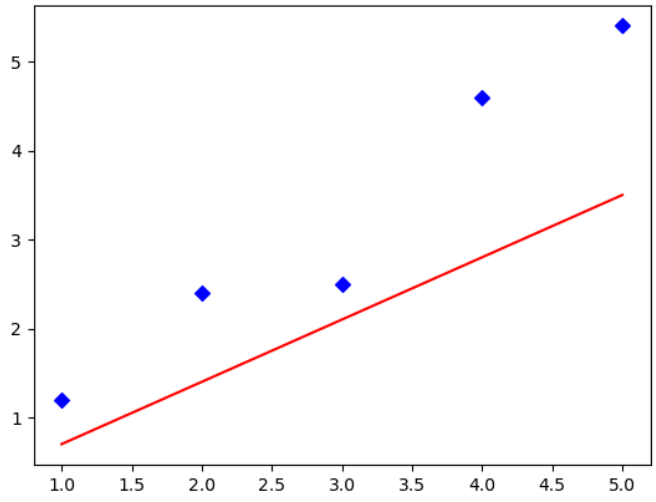
In [10]:
```
from sklearn import linear_model

regr = linear_model.LinearRegression()
# 남자는 0, 여자는 1
X = [[168, 0], [166, 0], [173, 0], [165, 0], [177, 0], [163, 0], W
     [178, 0], [172, 0], [163, 1], [162, 1], [171, 1], [162, 1], W
     [164, 1], [162, 1], [158, 1], [173, 1], ]    # 입력데이터를 2차원으로 만들어야 함
y = [65, 61, 68, 63, 68, 61, 76, 67, 55, 51, 59, 53, 61, 56, 44, 57]     # y 값은 1차원 데이터
regr.fit(X, y)          # 학습
print('계수 :', regr.coef_ )
print('절편 :', regr.intercept_)
print('점수 :', regr.score(X, y))
print('동윤이와 은지의 추정 몸무게 :', regr.predict([[167, 0], [167, 1]]))
```

```
계수 : [ 0.74803397 -7.23030041]
절편 : -61.227783894306384
점수 : 0.8425933302504424
동윤이와 은지의 추정 몸무게 : [63.69388959 56.46358918]
```

오차 함수 구현 및 파라미터 공간의 최적값

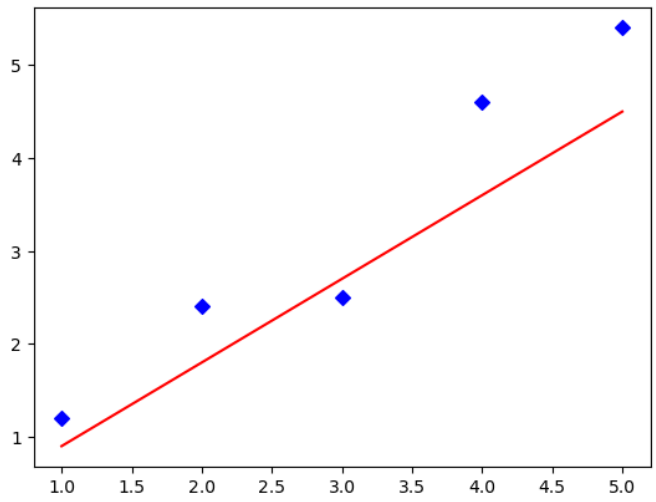
In [6]:
```
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5]) #
y = np.array([1.2, 2.4, 2.5, 4.6, 5.4]) # y 데이터

plt.scatter(x, y, color='blue', marker='D') #산점도를 알려줌
# 추정 그래프를 빨간색 실선으로 그림
plt.plot(x, 0.7 * x, 'r-')
```

Out[6]: [<matplotlib.lines.Line2D at 0x189df73c190>]



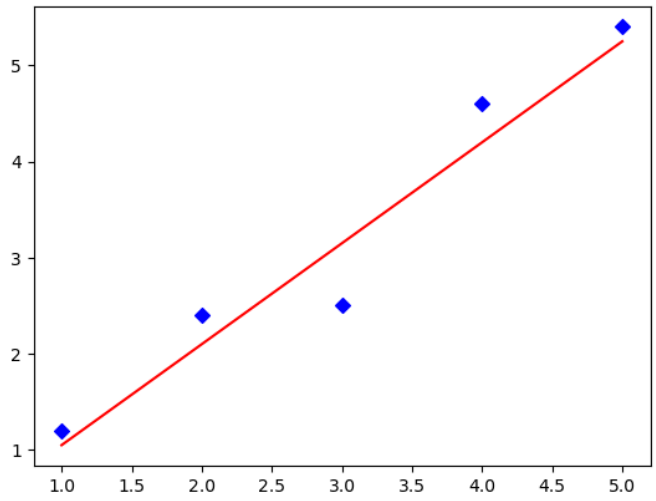
In [7]:
```
plt.scatter(x, y, color='blue', marker='D')
# 추정 그래프를 빨간색 실선으로 그림
plt.plot(x, 0.9 * x, 'r-')
```

Out[7]: [<matplotlib.lines.Line2D at 0x189df7b85e0>]



In [8]:
```
plt.scatter(x, y, color='blue', marker='D')
# 추정 그래프를 빨간색 실선으로 그림
plt.plot(x, 1.05 * x, 'r-')
```

Out[8]: [<matplotlib.lines.Line2D at 0x189df830e20>]



In [11]:
```
import numpy as np

y = np.array([1.2, 2.4, 2.5, 4.6, 5.4])
y_hat = np.array([1, 2, 3, 4, 5])
diff = (y_hat - y) ** 2     # y_hat과 y의 차이값의 제곱
e_mse = diff.sum() / len(y)
print('평균 제곱 오차 = ', e_mse)
```

평균 제곱 오차 = 0.19399999999999995

In [12]:
```
from sklearn.metrics import mean_squared_error

print('평균 제곱 오차 = ', mean_squared_error(y_hat, y))
```

평균 제곱 오차 = 0.19399999999999995